R E P U B L I Q U E F R A N Ç A I S E

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

BREVET D'INVENTION

CERTIFICAT D'UTILITÉ - CERTIFICAT D'ADDITION

COPIE OFFICIELLE

Le Directeur général de l'Institut national de la propriété industrielle certifie que le document ci-annexé est la copie certifiée conforme d'une demande de titre de propriété industrielle déposée à l'Institut.

Fait à Paris, le 22 JUIL. 2004

Pour le Directeur général de l'Institut
national de la propriété industrielle
Le Chef du Département des brevets

Martine PLANCHE

INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE
SIEGE
26 bis, rue de Saint-Petersbourg
75800 PARIS cedex 08
Téléphone : 33 (0)1 53 04 53 04
Télécopie : 33 (0)1 53 04 45 23
www.inpi.fr

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

BREVET D'INVENTION, CERTIFICAT D'UTILITÉ

Code de la propriété intellectuelle-Livre VI

cerfa N° 55-1328

REQUÊTE EN DÉLIVRANCE

26 bis, rue de Saint Pétersbourg
75800 Paris Cedex 08
Téléphone : 01 53 04 53 04 Télécopie : 01 42 93 59 30

Confirmation d'un dépôt par télécopie ☐

Cet imprimé est à remplir à l'encre noire en lettres capitales

Réservé à l'INPI

DATE DE REMISE DES PIÈCES 01 AVR 1999
N° D'ENREGISTREMENT NATIONAL 99 04072
DÉPARTEMENT DE DÉPÔT 75
DATE DE DÉPÔT 01 AVR. 1999

1 NOM ET ADRESSE DU DEMANDEUR OU DU MANDATAIRE À QUI LA CORRESPONDANCE DOIT ÊTRE ADRESSÉE

COMPAGNIE FINANCIERE ALCATEL
Département PI
Monsieur Sylvain CHAFFRAIX
30 avenue Kléber
75116 PARIS

n°du pouvoir permanent / références du correspondant / téléphone
PG 7176 F°101872PA/SYC 0140676300

2 DEMANDE Nature du titre de propriété industrielle

☒ brevet d'invention ☐ demande divisionnaire ☐ certificat d'utilité ☐ transformation d'une demande de brevet européen — demande initiale ☐ brevet d'invention ☐ certificat d'utilité n° date

Établissement du rapport de recherche ☐ différé ☒ immédiat

Le demandeur, personne physique, requiert le paiement échelonné de la redevance ☐ oui ☒ non

Titre de l'invention (200 caractères maximum)

PROCEDE DE MISE EN OEUVRE D'UNE ARBORESCENCE D'OBJETS DISTRIBUES

3 DEMANDEUR (S) n° SIREN 5 4 2 0 1 9 0 9 6 code APE-NAF

Nom et prénoms (souligner le nom patronymique) ou dénomination

ALCATEL

Forme juridique

Société anonyme

Nationalité (s) Française

Adresse (s) complète (s)

54 rue La Boétie
75008 PARIS

Pays

FRANCE

En cas d'insuffisance de place, poursuivre sur papier libre ☐

4 INVENTEUR (S) Les inventeurs sont les demandeurs ☐ oui ☒ non Si la réponse est non, fournir une désignation séparée

5 RÉDUCTION DU TAUX DES REDEVANCES ☐ requise pour la 1ère fois ☐ requise antérieurement au dépôt : joindre copie de la décision d'admission

6 DÉCLARATION DE PRIORITÉ OU REQUÊTE DU BÉNÉFICE DE LA DATE DE DÉPÔT D'UNE DEMANDE ANTÉRIEURE

pays d'origine	numéro	date de dépôt	nature de la demande

7 DIVISIONS antérieures à la présente demande n° date n° date

8 SIGNATURE DU DEMANDEUR OU DU MANDATAIRE (nom et qualité du signataire)

S. CHAFFRAIX / LC 40 B

SIGNATURE DU PRÉPOSÉ À LA RÉCEPTION SIGNATURE APRÈS ENREGISTREMENT DE LA DEMANDE À L'INPI

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

BREVET D'INVENTION, CERTIFICAT D'UTILITE

DÉSIGNATION DE L'INVENTEUR
(si le demandeur n'est pas l'inventeur ou l'unique inventeur)

DEPARTEMENT DES BREVETS

26bis, rue de Saint-Pétersbourg
75800 Paris Cédex 08
Tél. : 01 53 04 53 04 - Télécopie : 01 42 93 59 30

N° D'ENREGISTREMENT NATIONAL

9904072

F°101872PA - SYC/SSD

TITRE DE L'INVENTION :

PROCEDE DE MISE EN OEUVRE D'UNE ARBORESCENCE D'OBJETS DISTRIBUES

LE(S) SOUSSIGNÉ(S)

Société anonyme :
ALCATEL

DÉSIGNE(NT) EN TANT QU'INVENTEUR(S) (indiquer nom, prénoms, adresse et souligner le nom patronymique) :

- BANCTEL Fabrice) c/o ALCATEL CIT
- PIETRE Armel) Route de Nozay
) 91460 MARCOUSSIS, FRANCE

NOTA : A titre exceptionnel, le nom de l'inventeur peut être suivi de celui de la société à laquelle il appartient (société d'appartenance) lorsque celle-ci est différente de la société déposante ou titulaire.

Date et signature (s) du (des) demandeur(s) ou du mandataire
01.04.1999 PARIS

S. CHAFFRAIX

BA 113/140897

PROCEDE DE MISE EN ŒUVRE D'UNE ARBORESCENCE D'OBJETS DISTRIBUES

La présente invention concerne un procédé d'organisation hiérarchique d'objets distribués.

Cette invention s'applique à nombre d'applications utilisant un environnement d'objets distribués, comme, à titre d'exemple, les applications de supervision en télécommunication ou transport, les applications constituant un réseau intelligent...

Dans un environnement d'objets distribués, une application peut utiliser différents serveurs, pour fournir des services à des clients.

On appelle processus, un programme qui tourne dans un environnement donné. Un objet de ce processus est une entité logicielle dans ce processus.

Dans une application, les objets distribués sont en pratique organisés selon une arborescence donnée.

Dans cette arborescence, chaque objet à un nom logique, c'est à dire une chaîne de caractères, qui donne le chemin logique d'accès à cet objet depuis l'objet de départ, c'est à dire la racine, de l'arborescence. Ce nom logique est absolu, en ce sens qu'il est déterminé par rapport à la racine.

On peut aussi donner le chemin logique d'accès depuis un objet autre que la racine. On parle alors de nom logique relatif.

Dans tout système d'objets distribués basé sur un ORB, il est souvent nécessaire d'accéder directement à des objets. Pour cela, on doit utiliser les noms logiques, absolus ou relatifs de ces objets, permettant de trouver le chemin logique d'accès jusqu'à l'objet requis.

En outre, il est en général possible de demander directement à un objet parent d'accéder à un objet fils. Dans cette requête à l'objet parent, on utilise alors le nom logique relatif par rapport à cet objet parent pour désigner l'objet fils.

Or, comme les objets sont distribués, l'arborescence des objets comprend en pratique de nombreuses branches, et ces branches ou des parties de ces branches peuvent correspondre à des processus distincts. La figure 1 représente schématiquement un exemple simplifié d'une telle arborescence d'objets.

Sous un processus principal P0, qui constitue la racine de l'arborescence du système, on trouve trois processus distincts P1, P2, P3. Le processus P1 situé directement sous la racine, comprend trois objets, un premier objet A, qui est l'objet d'entrée ou racine du processus, duquel partent deux ramifications vers deux objets B et C. De l'objet B part une dernière ramification vers un objet D du processus P2.

Le processus P3 situé directement sous la racine, comprend un seul objet X.

Dans une mise en œuvre pratique de l'arborescence du système, ce sont les objets qui contiennent les informations sur leurs fils respectifs. Si ces objets fils sont contenus dans le même processus que l'objet parent, ces informations sont des pointeurs, donnant les adresses physiques de ces objets fils. Si ces objets fils ne sont pas contenus dans le même processus que l'objet parent, ces informations sont des références. Par exemple, l'objet A contient un pointeur sur l'objet B et un pointeur sur l'objet D. L'objet B contient lui une référence sur l'objet D.

Le chemin logique d'accès pour accéder l'objet D depuis la racine, c'est à dire le nom logique absolu, peut s'écrire /A/B/D. Il faut donc passer par l'objet B pour arriver sur l'objet D. Et ce, que l'on accède à

l'objet D, en interrogeant directement l'objet B, identifié par le nom logique absolu /A/B, pour l'objet fils identifié dans l'objet B par une référence ou en interrogeant la racine pour l'objet identifié par le nom logique (absolu) /A/B/D.

Or, les objets A et B sont dans le processus P1 et l'objet D est dans un autre processus P2. Si le processus P1 n'est pas en route, ou est en panne, il n'est plus possible selon cette implémentation de l'arborescence d'accéder l'objet D.

En outre, une telle implémentation ne permet pas une gestion aisée de la redondance des processus. Dans l'exemple représenté sur la figure 1, on a ainsi un processus redondant P2', de secours, prévu pour remplacer le processus P2 s'il tombe en panne. Avec la mise en œuvre de l'arborescence expliquée précédemment, c'est à chaque objet qui reçoit une demande sur un objet qui n'est pas dans son processus, de déterminer sur quel processus P2 ou P2' il va transmettre sa demande. On comprend que cela rend particulièrement complexe la gestion de la redondance. Or les processus redondants sont couramment utilisés pour renforcer les points faibles d'un système c'est à dire, les processus qui sont soit susceptibles de tomber souvent en panne, soit dont la panne paralyse l'ensemble du système ou simplement réduit la qualité du service.

Pour ces raisons, une autre implémentation de l'arborescence des objets d'un système d'objets distribués a été proposée, pour permettre l'accès à tous les objets du système, même si certains objets parents sont indisponibles (processus en panne ou arrêté) et pour simplifier la gestion de la redondance de processus. Dans cette implémentation, on a une gestion centralisée de l'arborescence au niveau de la racine, par un répertoire central, qui contient tous

les noms structurés de tous les objets. En d'autres termes, il contient toute l'arborescence du système.

Dans cette implémentation, si on interroge un objet parent pour un objet fils, l'appel est redirigé vers le répertoire central. On peut alors toujours accéder à un objet même si dans l'arborescence, cet objet dépend en filiation d'autres processus qui sont arrêtés. En outre la gestion de la redondance se trouve elle aussi centralisée, gérée par ce même répertoire central.

Cependant, cette implémentation est très coûteuse en termes de ressources : si le nombre d'objets est important, le répertoire central peut-être surchargé et les performances du système sérieusement dégradées, du fait du temps nécessaire pour consulter l'arborescence dans le répertoire central pour chaque appel.

En outre, cette solution ne prend plus en compte la spécificité de l'environnement distribué, puisqu'elle traite chaque objet de manière identique. Tous les appels sont traités par le répertoire central, même si l'appel concerne un objet fils d'un objet parent situé dans le même processus. Ceci augmente inutilement le volume de communication inter processus.

Enfin, si le système utilise comme protocole de communication entre objets, le protocole objet-objet, basé sur la création de paires d'éléments représentants, comme par exemple les paires proxy/stub dans les environnements distribués basés sur l'ORB DCOM, la solution à répertoire centralisé multiplie ces paires, puisqu'elle implique la création d'une paire d'éléments représentants pour chaque objet de l'arborescence. Or ces paires d'éléments représentants sont très coûteuses en termes de ressources mémoire.

Ainsi, l'invention a pour objet un procédé de mise en œuvre d'une arborescence d'objets distribués qui ne présentent pas les inconvénients précités.

Selon l'invention, on utilise un répertoire central qui ne contient des informations d'arborescences que sur certains objets ciblés, en sorte que tout objet d'un processus puisse être accédé.

Selon l'invention, lorsqu'un objet parent reçoit une demande de localisation d'un objet fils, il accède à l'objet fils, si ce dernier est dans le même processus, ou il retourne l'appel vers le répertoire central, s'il n'est pas dans le même processus.

En d'autres termes, l'arborescence à l'intérieur d'un même processus, est gérée en interne dans ce processus, les objets de ce processus contenant les pointeurs nécessaires sur les objets fils contenus dans ce processus, c'est à dire les adresses physiques de ces objets dans le processus considéré, mais l'arborescence des processus est gérée par le répertoire central. Cela permet avantageusement d'accéder à des objets de processus fils même si un processus parent est arrêté ; cela permet de gérer les problèmes de redondance au niveau du répertoire central; enfin, cela permet d'optimiser le temps de réponse du répertoire central qui n'a qu'une arborescence partielle à gérer et d'optimiser les ressources mémoires nécessaires pour implémenter cette arborescence.

Telle que caractérisée, l'invention concerne donc un procédé de mise en œuvre d'une arborescence d'objets distribués selon la revendication 1.

D'autres caractéristiques et avantages de l'invention sont décrits dans la description suivante, faite à titre indicatif et nullement limitatif et en référence aux dessins annexés dans lesquels :

- la figure 1 déjà décrite représente un schéma simplifié d'implémentation d'une arborescence d'objets distribués selon l'état de la technique; et

- la figure 2 représente un schéma d'une implémentation d'une arborescence d'objets distribués selon l'invention.

Selon l'invention, un répertoire central est prévu, correspondant au processus Pr0 sur la figure 2. Ce processus est la racine de l'arborescence.

Sous le processus racine Pr0, on trouve différents processus.

Un premier processus Pr1 contient trois objets A, B et C. Dans ce processus, l'objet A est l'objet racine. On appelle objet racine d'un processus, un objet d'entrée de ce processus. On remarquera qu'il peut y en avoir plusieurs dans un même processus.

Les objets B et C sont deux objets fils respectifs de l'objet A.

Un processus redondant Pr1' est la réplique de ce premier processus. Notamment, il contient les mêmes objets selon la même arborescence.

Un deuxième processus Pr2 contient deux objets D et F. Dans ce processus, l'objet D est la racine et l'objet F un objet fils de l'objet D. L'objet D est en outre objet fils de l'objet B du processus Pr1.

Un processus redondant Pr2' est la réplique de ce deuxième processus. Notamment, il contient les mêmes objets selon la même arborescence.

Le répertoire central contient une structure de données Tab0, dans laquelle il mémorise des informations relatives à l'arborescence du système.

En pratique il contient au moins toutes les informations relatives aux objets d'entrée, ou racine de chaque processus distinct de l'arborescence.

Dans l'exemple, à l'entrée E1 de la structure de données, on a des informations relatives à l'objet A du processus Pr1 : nom logique par rapport au répertoire central /A, pointeur pPr1 sur le processus

correspondant Pr1, et d'autres informations nécessaires à sa gestion.

A l'entrée E2, il trouve les informations concernant l'objet A du processus redondant Pr1' ; à l'entrée E3, celles sur l'objet D du processus Pr2 ; à l'entrée E4, celles sur l'objet D du processus Pr2'.

Ainsi, le répertoire central contient l'arborescence des processus dans le système.

Selon l'invention, un objet parent dans un processus (autre que le répertoire central) contient des informations sur ses objets fils qui sont des pointeurs, c'est à dire leur adresse physique, s'ils sont contenus dans le même processus. Ainsi, l'objet A contient un pointeur pB, respectivement pC sur l'objet fils B, respectivement C.

Dans le cas où l'objet fils n'est pas dans le même processus, l'objet parent contient une information pour retourner l'appel au répertoire central. Ainsi, si l'objet B reçoit une demande pour l'objet fils D identifié par son nom logique /D relatif par rapport à l'objet B, ce dernier renvoie la demande sur le répertoire central.

En pratique, il renvoie cette demande en plaçant la chaîne de caractères de son propre nom logique absolu, par rapport au répertoire central, devant la chaîne de caractères du nom logique relatif de l'objet D. Dans l'exemple, le nom logique absolu de l'objet B est égal à la chaîne de caractères /A/B. Ainsi, l'objet B transmet la demande au répertoire central en lui fournissant le nom logique absolu N(D)=/A/B/D de l'objet D.

Lorsque le répertoire central reçoit une demande sur un objet identifié par son nom logique par rapport au répertoire central, il consulte sa structure de données interne, de type dictionnaire, dans laquelle il recherche la chaîne de caractères correspondante. Si il

la trouve, il obtient une référence correspondante de l'objet dans le système. Cette référence lui permet de transmettre la demande directement sur cet objet. S'il ne la trouve pas, il recherche la chaîne de caractères la plus grande possible correspondant à une première partie de la chaîne de caractères, afin de transmettre la demande sur un objet parent pour l'objet donné identifié par son nom relatif par rapport à cet objet parent. Ce nom relatif est obtenu par la différence entre les deux chaînes de caractères. Prenons l'exemple d'une demande reçue par le répertoire pour l'objet C défini par son nom logique N(C)=/A/C .

Le répertoire central recherche dans sa structure de données cette chaîne ou une chaîne la plus grande possible correspondant à la première partie (c'est à dire le début de cette chaîne). Dans l'exemple, il va trouver la chaîne /A, qui est le nom logique de l'objet A.

Il transmet donc la demande sur l'objet C à l'objet A, en lui passant comme identifiant le nom logique relatif de l'objet C par rapport à cet objet A. Ce nom logique relatif est obtenu simplement par la différence entre les deux chaînes de caractères : /A/C - /A = /C.

Dans l'invention, on prévoit que si l'objet sur lequel le répertoire a transmis une demande sur un objet fils, ne trouve pas ce dernier dans son processus, il envoie un message au répertoire central, qui va rechercher un autre objet dans son répertoire. Il peut aussi mettre une information correspondante dans sa structure de données.

En ce qui concerne la gestion de la redondance, on voit sur la figure 2, que la structure de donnée Tab0 contient tous les objets de mêmes noms logiques correspondants à des processus différents. A chaque entrée dans la table, correspond une identification physique du processus correspondant. Ainsi, à l'entrée

E1, on trouve le nom /A pour un processus identifié par un paramètre pPr1, correspondant au processus Pr1. A l'entrée E2, on trouve le nom /A pour un processus identifié par un paramètre pPr1', correspondant au processus redondant Pr1'.

Comme dans l'invention, dès qu'un objet d'un processus a une demande sur un objet fils d'un autre processus à gérer, il transmet sa demande sur le répertoire central, c'est ce dernier qui assure toute la gestion de la redondance. En d'autres termes, c'est lui qui détermine à un moment donné s'il transmet l'appel sur le processus Pr1 ou sur son processus Pr1' selon des informations sur l'état du système. La gestion de la redondance s'en trouve centralisée.

De préférence, on a vu que le répertoire central contient les informations relatives aux objets d'entrée (racine) de chacun des processus du système. Il contient donc l'arborescence des processus, (y compris la redondance), tandis que l'arborescence dans les processus est implémentée en interne dans chacun de ces processus.

Enfin, on notera que le répertoire central est un point sensible du système. On prévoira donc en pratique des mécanismes de protection ou un répertoire central redondant afin d'obtenir un mécanisme robuste.

On a vu que l'invention s'applique dans un environnement d'objets distribués.

Une application particulière concerne un environnement basé sur un gestionnaire d'objets distribués ORB, acronyme anglo-saxon pour *Object Request Broker*. On peut citer comme ORB connu et utilisés, l'ORB CORBA (*Common Object Request Broker Architecture*) et DCOM (*Distributed Component Object Mode)*.

REVENDICATIONS

1. Procédé de mise en œuvre d'une arborescence d'objets distribués dans différents processus, un répertoire central (Pr0) apte à mémoriser des informations sur des objets dans une structure de données (Tab0) étant à la racine de l'arborescence, caractérisé en ce que pour chaque objet fils (B), un objet parent (A) dans un processus contient :

- une information correspondant à une adresse physique (pB) si l'objet fils est contenu dans ledit processus, et
- une information renvoyant au dit répertoire central, si l'objet fils n'est pas contenu dans le même processus.

2. Procédé selon la revendication 1, caractérisé en ce que lorsque le répertoire central (Pr0) reçoit une demande d'accès sur un premier objet (C) identifié par un nom logique identifiant un chemin logique d'accès du dit premier objet depuis le répertoire central (/A/C), il recherche dans sa structure de donnée le nom logique reçu, pour transmettre la demande directement sur le dit objet ou bien, si ce nom logique n'est pas dans son répertoire, il recherche un nom logique (/A) avec chaîne de caractères la plus grande possible égale à une première partie de la chaîne de caractères du nom logique reçu, pour transmettre sur un objet parent ainsi déterminé la demande sur le premier objet, en fournissant au dit objet parent une information (/B) correspondant au chemin logique d'accès du premier objet par rapport à l'objet parent.

3. Procédé selon la revendication 2, caractérisé en ce que l'objet parent qui reçoit ladite demande, transmet la demande sur le dit premier objet, si c'est un objet fils de son processus, ou retourne un message au répertoire central.

4. Procédé selon l'une quelconque des revendications précédentes, caractérisé en ce que le répertoire central assure la gestion de la redondance des processus, par la sélection d'un processus parmi plusieurs possibles contenant l'objet demandé.

5. Procédé selon l'une quelconque des revendications précédentes, caractérisé en ce que lorsqu'un objet parent d'un processus reçoit directement une demande sur un objet fils, il renvoie cette demande sur le répertoire central, si le dit objet fils n'est pas contenu dans son processus.

6. Procédé selon la revendication 5, l'objet fils étant identifié dans ladite demande par un nom logique définissant le chemin logique d'accès de cet objet depuis ledit objet parent, caractérisé en ce que ledit objet parent renvoie ladite demande au répertoire central en faisant précéder la chaîne de caractères de ce nom logique par la chaîne de caractères correspondant à son propre nom logique définissant son chemin logique d'accès depuis le répertoire central.

7. Procédé selon l'une quelconque des revendications précédentes, caractérisé en ce que le répertoire central contient au moins des informations relatives à chaque objet racine de chaque processus.

8. Procédé selon l'une quelconque des revendications précédentes, caractérisé en ce qu'il s'applique à un environnement des objets distribués basé sur un gestionnaire de type CORBA.

9. Procédé selon l'une quelconque des revendications 1 à 7, caractérisé en ce qu'il s'applique à un environnement des objets distribués basé sur un gestionnaire de type DCOM.

FIG_1

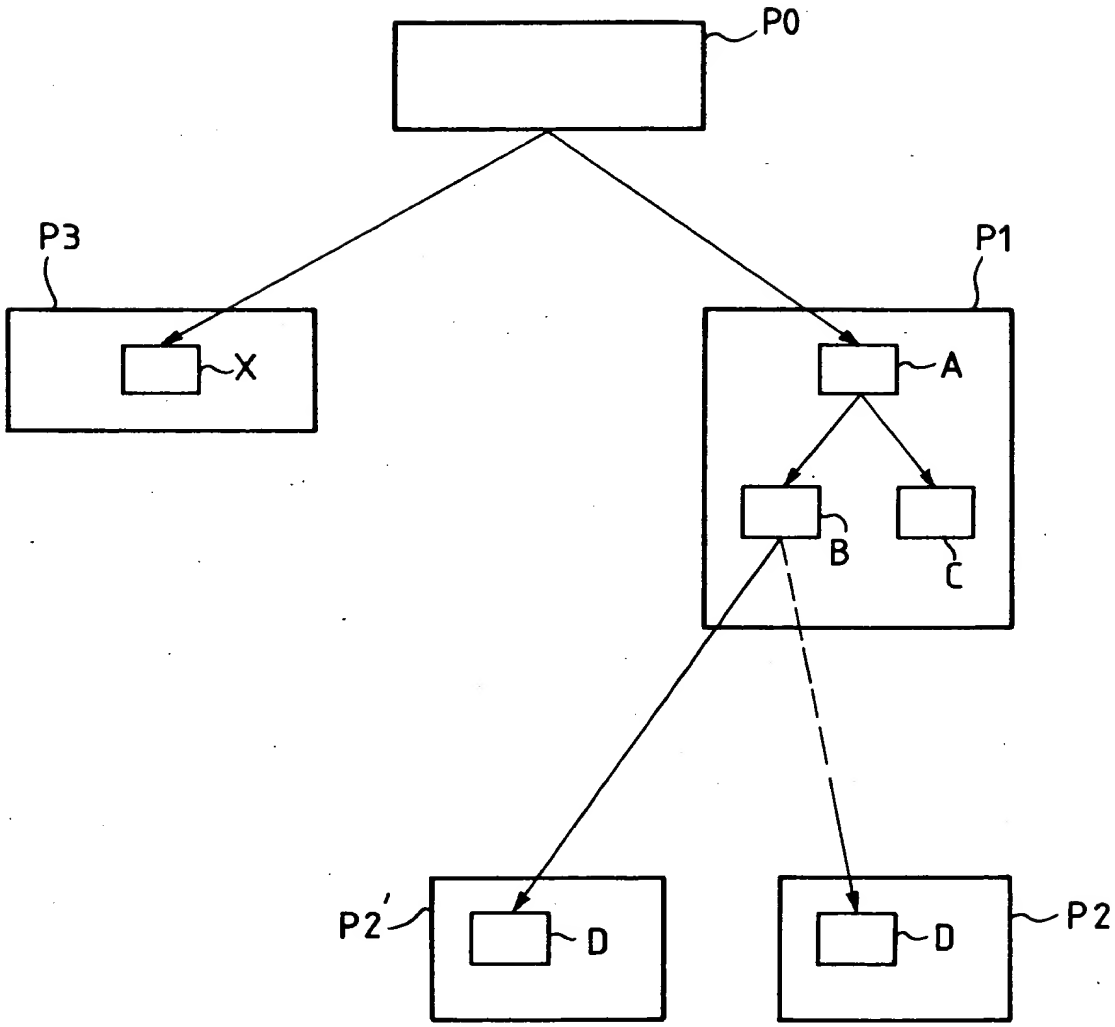
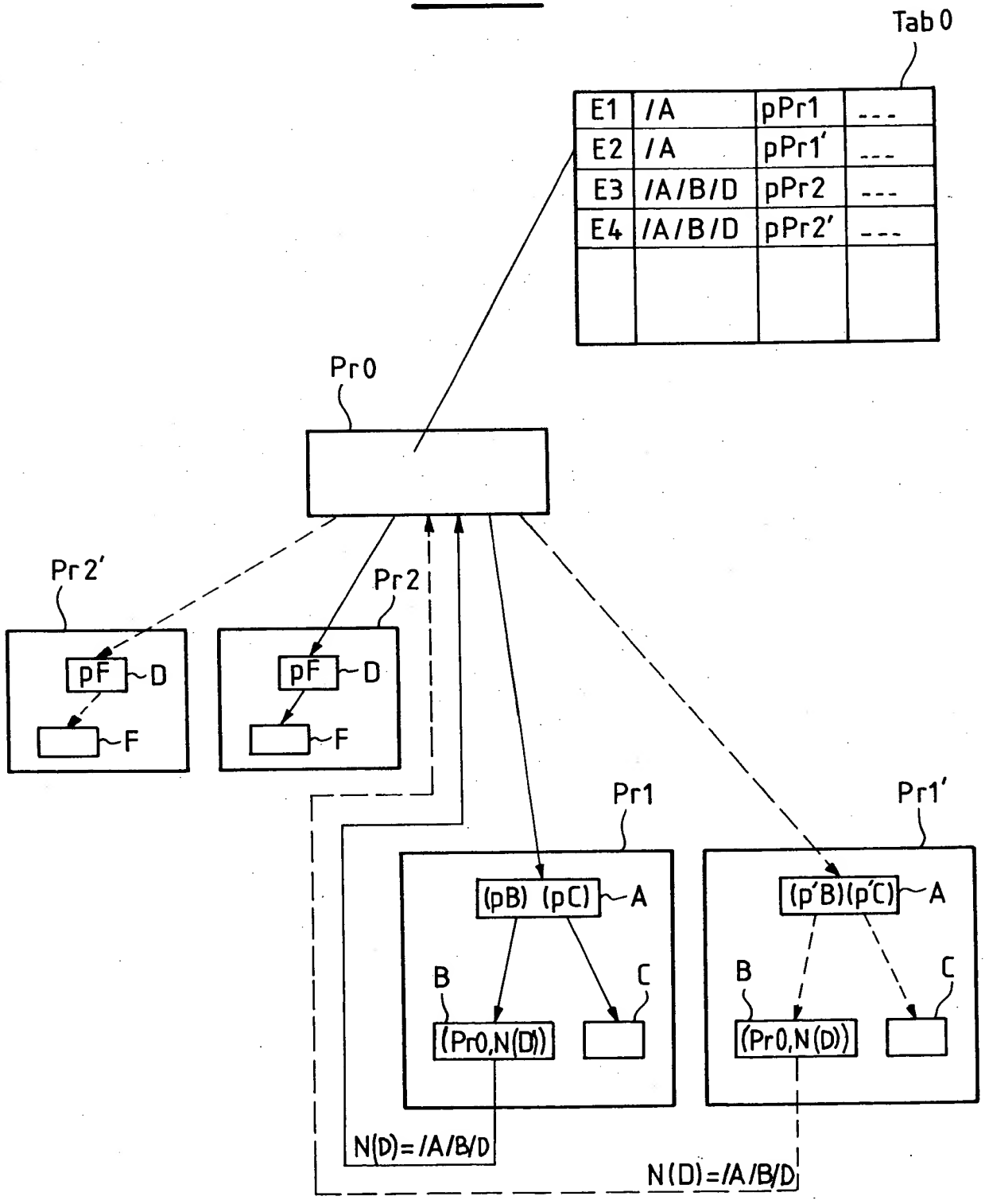


FIG_2


Tab 0
E1 /A pPr1 ---
E2 /A pPr1' ---
E3 /A/B/D pPr2 ---
E4 /A/B/D pPr2' ---
Pr0
Pr2'
Pr2
pF D
F
pF D
F
Pr1
Pr1'
(pB) (pC) A
B
C
(Pr0,N(D))
(p'B)(p'C) A
B
C
(Pr0,N(D))
N(D)=/A/B/D
N(D)=/A/B/D

UNITED STATES PATENT AND TRADEMARK OFFICE

I, Susan ANTHONY BA, ACIS,

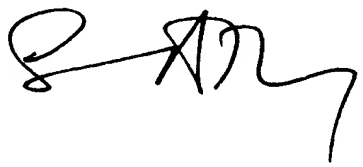
Director of RWS Group Ltd, of Europa House, Marsham Way, Gerrards Cross, Buckinghamshire, England declare;

1. That I am a citizen of the United Kingdom of Great Britain and Northern Ireland.

2. That the translator responsible for the attached translation is well acquainted with the French and English languages.

3. That the attached is, to the best of RWS Group Ltd knowledge and belief, a true translation into the English language of the accompanying copy of the specification filed with the application for a patent in France on April 1, 1999 under the number 99/04,072 and the official certificate attached hereto.

4. That I believe that all statements made herein of my own knowledge are true and that all statements made on information and belief are true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the patent application in the United States of America or any patent issuing thereon.



For and on behalf of RWS Group Ltd

The 20th day of July 2004

FRENCH REPUBLIC

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

PATENT

UTILITY CERTIFICATE – CERTIFICATE OF ADDITION

OFFICIAL COPY

The Director-General of the Institut National de la Propriété Industrielle certifies that the attached document is a true copy of an application for industrial property titleright filed at the Institute.

Drawn up in Paris, **05 APR. 2000**

On behalf of the Director-General of the
Institut National de la Propriété Industrielle
The Patent Department Head

[signature]

Martine PLANCHE

INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE
REGISTERED OFFICE
26 bis, rue de Saint Petersbourg
75800 PARIS Cédex 08
Telephone: 01 53 04 53 04
Fax: 01 42 93 59 30

NATIONAL PUBLIC ESTABLISHMENT CREATED BY LAW No. 51-444 OF 19 APRIL 1951

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

PATENT, UTILITY CERTIFICATE

Intellectual Property Code - Book VI

Cerfa
No. 55-1328

REQUEST FOR GRANT

26 bis, rue de Saint Pétersbourg
75800 Paris Cedex 08
Telephone: 01 53 04 53 04 Fax: 01 42 93 59 30

Confirmation of filing by fax ☐

This form is to be completed in black ink and in block capitals

Reserved for the INPI	
DATE OF SUBMISSION OF THE DOCUMENTS	01. APR. 1999
NATIONAL REGISTRATION	99/04,072
DEPARTMENT OF FILING	75
DATE OF FILING	01 APR. 1999

1. NAME AND ADDRESS OF THE APPLICANT OR THE REPRESENTATIVE TO WHOM THE CORRESPONDENCE IS TO BE ADDRESSED

COMPAGNIE FINANCIERE ALCATEL
Département PI
Mr Sylvain CHAFFRAIX
30 avenue Kléber
75116 PARIS

No. of permanent power of attorney	Correspondent's references	Telephone
PG 7176	F°101872PA/SYC	0140676300

2. APPLICATION Nature of the industrial property right

☒ patent ☐ divisional application → initial application ↓

☐ utility certificate ☐ conversion of a European patent application ☐ patent ☐ utility certificate No. date

Compilation of the search report ☐ deferred ☒ immediate

The applicant, as a physical person, asks to pay the fee by instalments ☐ yes ☒ no

Title of the invention (maximum 200 characters)

METHOD OF IMPLEMENTING A TREE OF DISTRIBUTED OBJECTS

3. APPLICANT(S) SIREN No. 5 4 2 0 1 9 0 9 6 APE-NAF code

Name and forenames (underline the surname) or company name

ALCATEL

Legal form

Société Anonyme

Nationality/Nationalities French

Full address(es)

54 rue La Boétie
75008 PARIS

Country

FRANCE

If insufficient space, continue on plain paper ☐

4. INVENTOR(S) The inventors are the applicants ☐ yes ☒ no If the answer is no, provide a separate designation

5. REDUCTION OF THE RATE OF FEES ☐ requested for the first time ☐ requested prior to filing; attach copy of the favourable decision

6. PRIORITY DECLARATION OR APPLICATION FOR THE BENEFIT OF THE FILING DATE OF A PRIOR APPLICATION

Country of origin	Number	Filing date	Nature of the application

7. DIVISIONS previous to the present application No. date No. date

8. SIGNATURE OF THE XXXXXXXXX REPRESENTATIVE (name and capacity of the signatory - registration No.)

[signature]

S. CHAFFRAIX / LC 40 B

SIGNATURE OF THE RECEIVING OFFICIAL

SIGNATURE AFTER REGISTRATION OF THE APPLICATION AT THE INPI

[illegible signature]

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

PATENT, UTILITY CERTIFICATE

DESIGNATION OF THE INVENTOR
(if the applicant is not the inventor or the sole inventor)

PATENT DEPARTMENT

26bis, rue de Saint-Petersbourg
75800 Paris Cédex 08
Tel: 01 53 04 53 04 - Fax: 01 42 93 59 30

NATIONAL REGISTRATION NO.

99/04,072

F°101872PA – SYC/SSD

TITLE OF THE INVENTION:

METHOD OF IMPLEMENTING A TREE OF DISTRIBUTED OBJECTS

THE UNDERSIGNED

Société Anonyme:
ALCATEL

DESIGNATE(S) AS INVENTOR(S) (surname underlined, forenames, address):

- BANCTEL Fabrice) c/o ALCATEL CIT
) Route de Nozay
- PIETRE Armel) 91460 MARCOUSSIS, FRANCE

NOTE: In exceptional cases, the name of the inventor may be followed by that of the company to which he belongs (membership company) when the latter is other than the company which is the applicant or proprietor.

Date and signature(s) XXXXXXXXXX of the representative
1.4.1999 PARIS

[signature]
S. CHAFFRAIX

METHOD OF IMPLEMENTING A TREE OF DISTRIBUTED OBJECTS

The present invention concerns a method of organizing distributed objects in a hierarchy.

The invention applies to many applications employing a distributed object environment, for example telecommunication or transportation supervision applications, applications constituting an intelligent network, etc.

In a distributed object environment, an application can use different servers to provide services to clients.

The term "process" refers to a program which runs in a given environment. An object of the process is a software entity in the process.

In an application, the distributed objects are in practice organized in accordance with a given tree.

In the tree, each object has a logical name, i.e. a character string, which specifies the logical access path to that object from the starting object, i.e. the root, of the tree. The logical name is absolute in the sense that it is determined relative to the root.

It is also possible to specify the logical access path from an object other than the root. The term "relative logical name" is then used.

In any distributed object system based on an ORB, it is often necessary to access objects directly. The absolute or relative logical names of the objects must be used for this, enabling a logical access path to the required object to be found.

Furthermore, it is generally possible to ask a father object directly for access to a son object. In the request to the father object the relative logical name relative to the father object is used to designate the son object.

Because the objects are distributed, the tree of the objects in practice has many branches and the branches or parts of the branches can correspond to distinct processes. Figure 1 is a diagram showing a simplified example of an object tree of this kind.

Under a main process P0, which constitutes the root of the tree of the system, there are three distinct processes P1, P2, P3. The process P1 situated directly under the root comprises three objects, namely a first object A, which is the entry object or root of the process, from which depart two branches to two objects B and C. A final branch departs from the object B towards an object D of the process P2.

The process P3 situated directly under the root comprises a single object X.

In a practical implementation of the tree of the system the objects contain information on their respective sons. If the son objects are contained in the same process as the father object, the information consists of pointers giving the physical addresses of the son objects. If the son objects are not contained in the same process as the father object, the information is made up of references. For example, the object A contains a pointer to the object B and a pointer to the object D. The object B contains a reference to the object D.

The logical access path for accessing the object D from the root, i.e. the absolute logical name, can be written /A/B/D. It is therefore necessary to pass through the object B to arrive at the object D. This is the case whether the object D is reached by interrogating the object B directly, that object being identified by the absolute logical name /A/B, for the son object identified in the object B by a reference, or by interrogating the root for the object identified by the (absolute) logical name /A/B/D.

The objects A and B are in the process P1 and the object D is in another process P2. If the process P1 is not running, or has failed, it is no longer possible to access the object D with this tree.

Furthermore, this tree makes managing process redundancy difficult. In the example shown in figure 1, there is a redundant back-up process P2' which is substituted for the process P2 if it fails. With the tree previously explained, it is up to each object that receives a request regarding an object that is not in its process to determine to which of the two processes P2 and P2' it should transmit the request. Clearly this makes managing redundancy particularly complex. Redundant processes are routinely used to strengthen the weak points of a system, i.e. the processes that are likely to fail often, whether their failure paralyses the entire system or merely reduces the quality of service.

For the above reasons a different tree of the objects of a distributed object system has been proposed, to enable access to all the objects of the system even if some father objects are not available (process failed or stopped) and to simplify the management of process redundancy. This tree employs centralized management of the tree at the level of the root, by means of a central directory, which contains structured names of all the objects. In other words, it contains all of the tree of the system.

In this tree, if a father object is interrogated for a son object, the call is

redirected to the central directory. It is then always possible to access an object even if, within the tree, that object is a son of other processes which are stopped. What is more, this also centralizes management of redundancy, which is managed by means of the same central directory.

However, this tree is very costly in terms of resources: if the number of objects is large, the central directory can be overloaded and the performance of the system seriously degraded, because of the time needed to consult the tree in the central directory for each call.

Furthermore, this solution no longer takes into account the specific nature of the distributed environment, since it treats each object identically. All calls are processed by the central directory, even if the call concerns a son object of a father object in the same process. This increases the volume of inter-process communication unnecessarily.

Finally, if the system uses the object-object protocol based on the creation of pairs of representative elements, for example the proxy/stub pairs of distributed environments based on the DCOM ORB, as the protocol for communication between objects, the centralized directory solution multiplies the number of these pairs, because it implies the creation of a pair of representative elements for each object in the tree. The pairs of representative elements are very costly in terms of memory resources.

Accordingly, an object of the invention is to provide a method of implementing a tree of distributed objects that does not have the aforementioned disadvantages.

The invention uses a central directory which contains tree information on only certain targeted objects, and so all the objects of a process can be accessed.

According to the invention, if a father object receives a location request in respect of a son object, it accesses the son object, if the latter is in the same process, or returns the call to the central directory, if it is not in the same process.

In other words, the tree within a given process is managed internally in that process, the objects of the process containing the necessary pointers to the son objects contained in the process, i.e. the physical addresses of those objects in the process concerned, but the tree of processes is managed by the central directory. This has the advantage of providing access to objects of son processes even if a father process is stopped; this enables problems of redundancy at the level of the central directory to be managed; finally, it enables the response time of the central

directory to be optimized, because it has only a partial tree to manage, and it enables the memory resources necessary for implementing the tree to be optimized.

As characterized, the invention therefore concerns a method of implementing a tree of distributed objects according to claim 1.

Other features and advantages of the invention are described in the following description, which is given by way of non-limiting illustrative example only, and with reference to the accompanying drawings, in which:

- figure 1, already described, is a simplified block diagram of a prior art tree of distributed objects; and

- figure 2 is a block diagram of a tree of distributed objects in accordance with the invention.

The invention provides a central directory corresponding to the process Pr0 in figure 2. That process is the root of the tree.

There are different processes under the root process Pr0.

A first process Pr1 contains three objects A, B and C. In this process, the object A is the root object. The root object of a process is an entry object of the process. Note that there can be more than one in the same process.

The objects B and C are two son objects of the object A.

A redundant process Pr1' is a replica of the first process. In particular, it contains the same objects with the same tree.

A second process Pr2 contains two objects D and F. In this process, the object D is the root and the object F is a son object of the object D. The object D is also a son object of the object B of the process Pr1.

A redundant process Pr2' is a replica of the second process. In particular, it contains the same objects with the same tree.

The central directory contains a data structure Tab0 in which it stores information relating to the tree of the system.

In practice, it contains at least all the information relating to the entry objects (root objects) of each distinct process of the tree.

In this example, in entry E1 of the data structure there is information relating to the object A of the process Pr1: logical name relative to the central directory /A, pointer pPr1 to the corresponding process Pr1, and other information required for its management.

In entry E2 there is information concerning the object A of the redundant process Pr1'; in entry E3 there is information on the object D of the process Pr2; in

entry E4 there is information on the object D of the process Pr2'.

The central directory therefore contains the tree of the processes in the system.

According to the invention, a father object in a process (other than the central directory) contains information on its son objects, which takes the form of pointers, i.e. their physical addresses, if they are contained in the same process. Thus the object A contains pointers pB and pC to the son objects B and C, respectively.

If the son object is not in the same process, the father object contains information for returning the call to the central directory. Thus if the object B receives a request for the son object D identified by its logical name /D relative to the object B, the object B returns the request to the central directory.

In practice, it returns the request by placing the character string of its own absolute logical name, relative to the central directory, in front of the character string of the relative logical name of the object D. In the example, the absolute logical name of the object B is the character string /A/B. Accordingly, the object B transmits the request to the central directory by supplying it with the absolute logical name N(D)=/A/B/D of the object D.

If the central directory receives a request relating to an object identified by its logical name relative to the central directory, it consults its internal data structure, which is of the dictionary type, and looks up the corresponding character string. If it finds it, it obtains a corresponding reference of the object in the system. That reference enables it to transmit the request directly to the object. If it does not find it, it looks for the longest character string corresponding to a first part of the character string, in order to transmit the request to a father object for the given object identified by its relative name relative to that father object. That relative name is obtained as the reference between the two character strings. Take the example of a request received by the directory for the object C defined by its logical name N(C)=/A/C.

The central directory searches its data structure for that string or the longest string corresponding to its first part (i.e. the start of the string). In this example it will find the string /A, which is the logical name of the object A.

It therefore transmits the request relating to the object C to the object A, sending it by way of identification the relative logical name of the object C relative to the object A. That relative logical name is simply obtained as the difference between the two character strings: /A/C - /A = /C.

In accordance with the invention, if the object to which the directory has sent

a request relating to a son object cannot find the son object in its process, it sends a message to the central directory, which looks for another object in its directory. It can also place corresponding information in its data structure.

With regard to managing redundancy, figure 2 shows that the data structure Tab0 contains all the objects with the same logical names corresponding to different processes. To each entry in the table there corresponds a physical identification of the corresponding process. Thus in entry E1 there is the name /A for a process identified by a parameter pPr1 corresponding to the process Pr1. In the entry E2 there is the name /A for a process identified by a parameter pPr1' corresponding to the redundant process Pr1'.

As in the invention, as soon as an object of a process has a request relating to a son object of another process to manage, it sends its request to the central directory, which is entirely responsible for managing redundancy. In other words, it is the central directory which determines at a given time whether to send the call to the process Pr1 or to its process Pr1', depending on information on the status of the system. Thus management of redundancy is centralized.

As already mentioned, the central directory preferably contains information relating to the entry objects (root objects) of each process of the system. It therefore contains the tree of the processes (including the redundancy), while the tree in the processes is implemented internally in each process.

Finally, note that the central directory is a sensitive point of the system. Thus in practice protection mechanisms or a redundant central directory are provided in order to obtain a robust mechanism.

It has been shown that the invention applies in a distributed object environment.

One particular application concerns an environment based on a distributed Object Request Broker (ORB). ORBs known in the art include the CORBA (Common Object Request Broker Architecture) and DCOM (Distributed Component Object Mode) ORBs.

CLAIMS

1. A method of implementing a tree of distributed objects in different processes, there being a central directory (Pr0) adapted to store information on objects in a data structure (Tab0) at the root of the tree, characterized in that, for each son object (B), a father object (A) in a process comprises:

   - information corresponding to a physical address (pB) if the son object is contained in same process, or

   - information referring back to said central directory if the son object is not contained in the same process.

2. A method according to claim 1, characterized in that if the central directory (Pr0) receives a request for access to a first object (C) identified by a logical name identifying a logical access path of said first object from the central directory (/A/C), it searches its data structure for the logical name received in order to send the request directly to said object or, if said logical name is not in its directory, it searches for a logical name (/A) with the longest character string equal to a first part of the character string of the logical name received, in order to send to a father object determined in this way the request relating to the first object, by providing said father object with information (/B) corresponding to the logical access path of the first object relative to the father object.

3. A method according to claim 2, characterized in that the father object which receives said request sends the request to said first object if it is a son object of its process or returns a message to the central directory.

4. A method according to any preceding claim, characterized in that the central directory manages the redundancy of the processes by selecting one of several processes containing the requested object.

5. A method according to any preceding claim, characterized in that if a father object of a process receives a request relating to a son object directly it returns that request to the central directory if said son object is not contained in its process.

6. A method according to claim 5, the son object being identified in said request by a logical name defining the logical access path of that object from said father object, characterized in that said father object returns said request to the central directory with the character string of said logical name preceded by the character string corresponding to its own logical name defining its logical access path from the central directory.

7. A method according to any preceding claim, characterized in that the central directory contains at least information relating to each root object of each process.

8. A method according to any preceding claim, characterized in that it applies to a distributed object environment based on a manager of the CORBA type.

9. A method according to any of claims 1 to 7, characterized in that it applies to a distributed object environment based on a manager of the DCOM type.

FIG_1

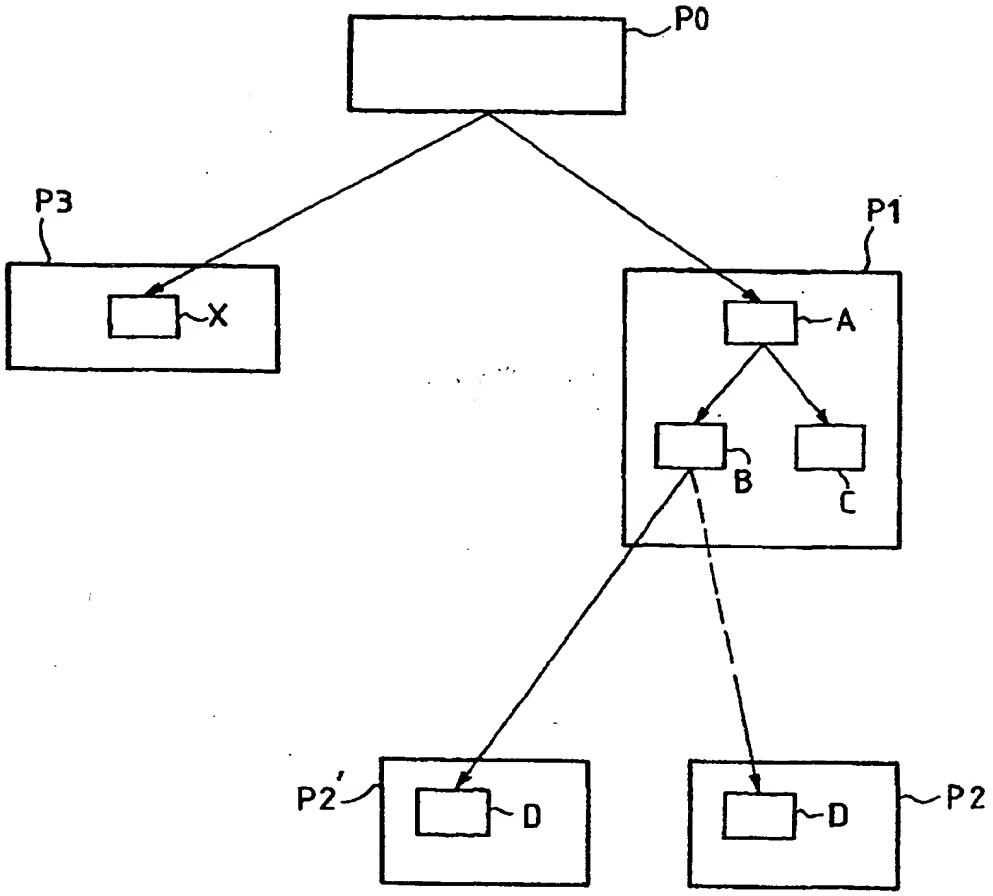
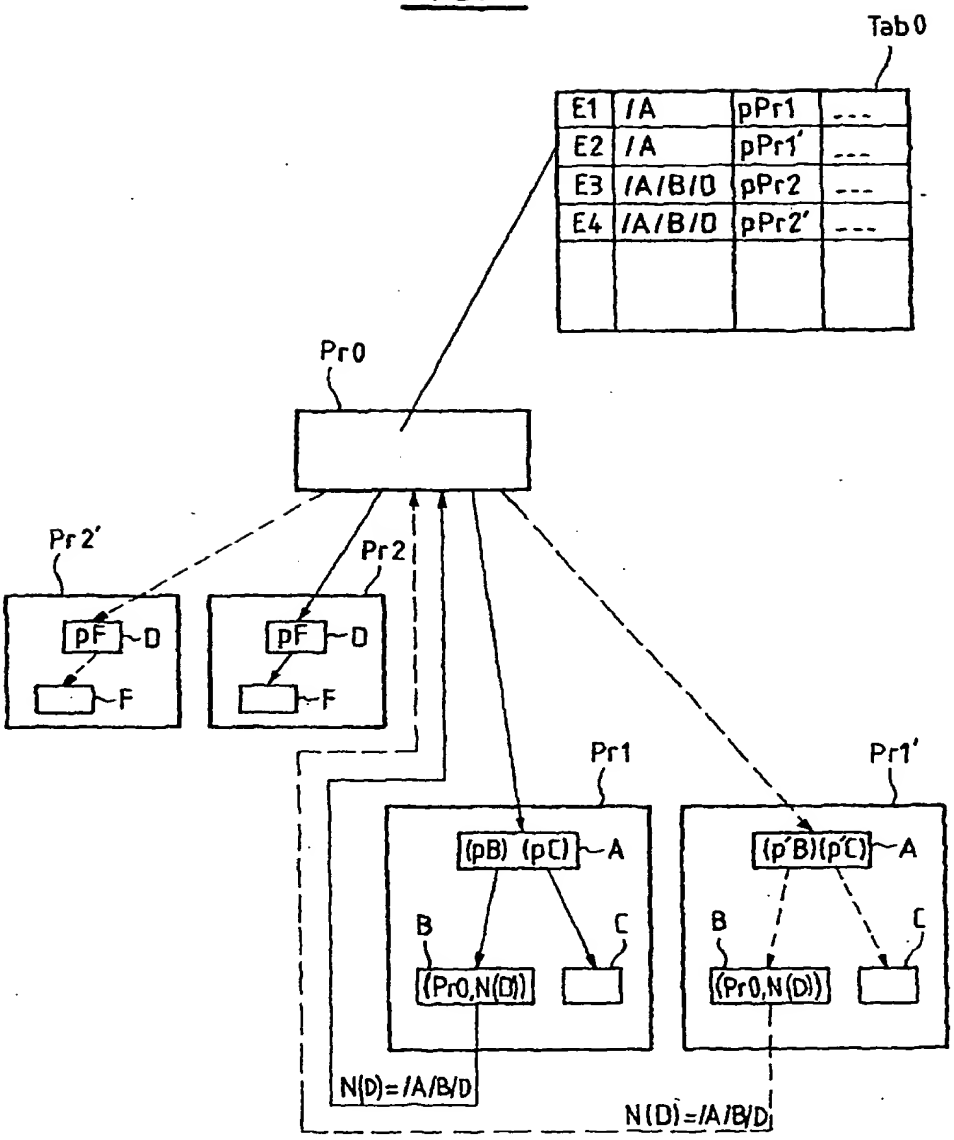


FIG_2


Tab 0
E1 /A pPr1 ---
E2 /A pPr1' ---
E3 /A/B/D pPr2 ---
E4 /A/B/D pPr2' ---
Pr0
Pr2'
Pr2
pF D
F
pF D
F
Pr1
Pr1'
(pB) (pC) A
B
C
(Pr0,N(D))
(p'B)(p'C) A
B
C
(Pr0,N(D))
N(D)=/A/B/D
N(D)=/A/B/D